Reliable Linux Wireless - Techniques for Debugging Wireless Module Integrations

STEVE DEROSIER / CAL-SIERRA CONSULTING

Introduction

- 50 minutes?!
- How to work with WiFi modules with the Linux kernel - especially misbehaving ones!
- Overview of Linux WiFi basic concepts - chips and stack
- Interfacing of devices
- Debugging tools
- HOW TO GET HELP!

CSC
CAL-SIERRA
CONSULTING

Why Steve deRosier?

- Steve deRosier - Principle Consultant at Cal-Sierra Consulting LLC
  - 10 years working with WiFi and the Linux-Wireless community
  - 17 years working with Embedded Linux
  - Contributed to OSS in many projects: binutils, buildroot, linux-wireless, linux-mtd, ALSA, others…
  - Linux-wireless: ath6kl, libertas_tf, o11s, brcmfmac, others…
  - Special knowledge of Laird's Linux wireless platforms

CSC
CAL-SIERRA
CONSULTING

WiFi Chips

- Full-MAC vs Soft-MAC
- Firmware
- Interfaces
- Other pins
- BT coexistence

CAL-SIERRA CONSULTING

WiFi Chips

- Full-MAC vs Soft-MAC
- Firmware
- Interfaces
- Other pins
- BT coexistence

- Chips fall into two major categories: full- and soft-MAC
- Primary difference is where upper-level logic resides
  - Full: in chip's firmware
  - Soft-MAC: in Linux's mac80211
- Not better/worse, just different tradeoffs

CSC
CAL-SIERRA
CONSULTING

WiFi Chips

- Full-MAC vs Soft-MAC
- Firmware
- Interfaces
- Other pins
- BT coexistence

- Most chips must load firmware
- Even soft-MAC chips have firmware
- Blobs from vendors, typically in /lib/firmware
- Linux-firmware git
- Load as late as possible to avoid delays or boot issues

CSC
CAL-SIERRA
CONSULTING

WiFi Chips

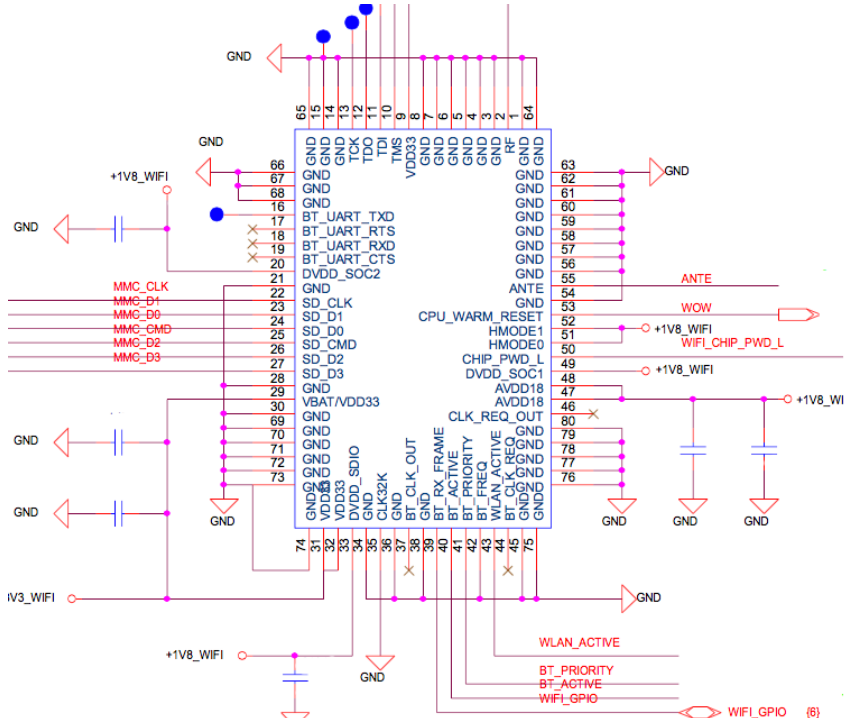
- Full-MAC vs Soft-MAC
- Firmware
- Interfaces
- Other pins
- BT coexistence

- Virtually every hardware bus interface is represented
  - SDIO
  - USB
  - PCIe
- Most drivers have multiple h/w interfaces
- h/w bus driver handles the bus; use the API but don't handle it directly
- Usually a bus-neutral abstraction layer for messaging

CSC
CAL-SIERRA
CONSULTING

WiFi Chips

- Full-MAC vs Soft-MAC
- Firmware
- Interfaces
- Other pins
- BT coexistence



CAL-SIERRA CONSULTING

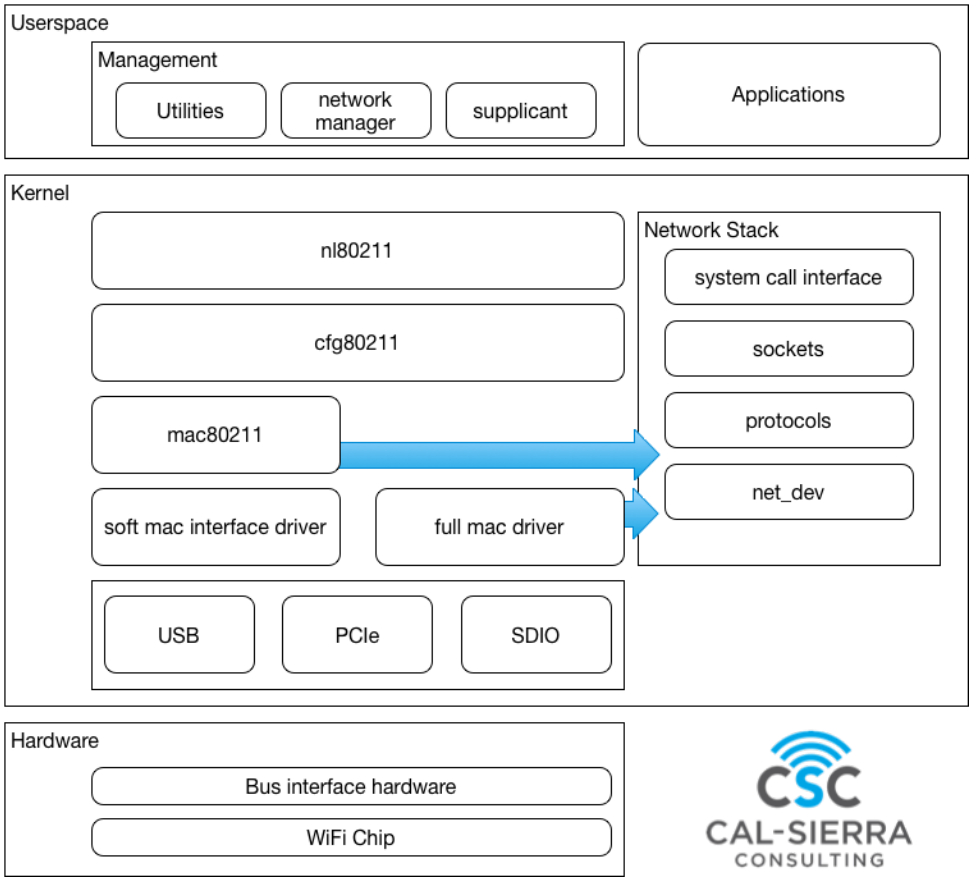
WiFi Chips

- Full-MAC vs Soft-MAC
- Firmware
- Interfaces
- Other pins
- BT coexistence

- BT integrated in some chips, not in others
- Pins for for BT, often UART
- Pins for coex
- Often ignored; needs setup

CAL-SIERRA CONSULTING

Linux WiFi Network Stack



CSC
CAL-SIERRA
CONSULTING

Linux wireless stack (from bottom up)

- Chip and bus
- Bus driver
- WiFi device driver
- mac80211 (Soft-MAC)
- cfg80211
- nl80211
- User-space management
- Data

CAL-SIERRA CONSULTING

Linux wireless stack (from bottom up)

- Chip and bus
- Bus driver
- WiFi device driver
- mac80211 (Soft-MAC)
- cfg80211
- nl80211
- User-space management
- Data

- Two types: Full and Soft-MAC
- Bus interfaces
  - SDIO
  - USB
  - PCIe

CSC
CAL-SIERRA
CONSULTING

Linux wireless stack (from bottom up)

- Chip and bus
- Bus driver
- WiFi device driver
- mac80211 (Soft-MAC)
- cfg80211
- nl80211
- User-space management
- Data

- Depends on the bus
- Uses standard Linux bus driver
- Types
  - SDIO
  - USB
  - PCIe
  - others

CAL-SIERRA CONSULTING

Linux wireless stack (from bottom up)

- Chip and bus
- Bus driver
- WiFi device driver
- mac80211 (Soft-MAC)
- cfg80211
- nl80211
- User-space management
- Data

- Two types, Full and Soft MAC
- Soft uses mac80211
- Uses the bus driver to talk to the hardware
- Usually a bus-independent abstraction layer to support multiple buses (eg ath6kl_core + ath6kl_sdio || ath6kl_usb)

CSC
CAL-SIERRA
CONSULTING

Linux wireless stack (from bottom up)

- Chip and bus
- Bus driver
- WiFi device driver
- mac80211 (Soft-MAC)
- cfg80211
- nl80211
- User-space management
- Data

- Linux WiFi MAC driver
- Used by Soft-MAC devices
- Use:
  - ieee80211_ops
  - ieee80211_alloc_hw
  - ieee80211_register_hw

CSC
CAL-SIERRA
CONSULTING

Linux wireless stack (from bottom up)

- Chip and bus
- Bus driver
- WiFi device driver
- mac80211 (Soft-MAC)
- cfg80211
- nl80211
- User-space management
- Data

- Main configuration API used
- All drivers (Full and SoftMAC) use it
- Replaced wext

CSC
CAL-SIERRA
CONSULTING

Linux wireless stack (from bottom up)

- Chip and bus
- Bus driver
- WiFi device driver
- mac80211 (Soft-MAC)
- cfg80211
- nl80211
- User-space management
- Data

- Userspace interface to cfg80211
- Along with cfg80211 replaces wext
- No more ioctls

CSC
CAL-SIERRA
CONSULTING

Linux wireless stack (from bottom up)

- Chip and bus
- Bus driver
- WiFi device driver
- mac80211 (Soft-MAC)
- cfg80211
- nl80211
- User-space management
- Data

- Basic tools like `ip`, `iw` and so on
- wpa_supplicant
- Higher level tools like NetworkManager, conman, etc

CSC
CAL-SIERRA
CONSULTING

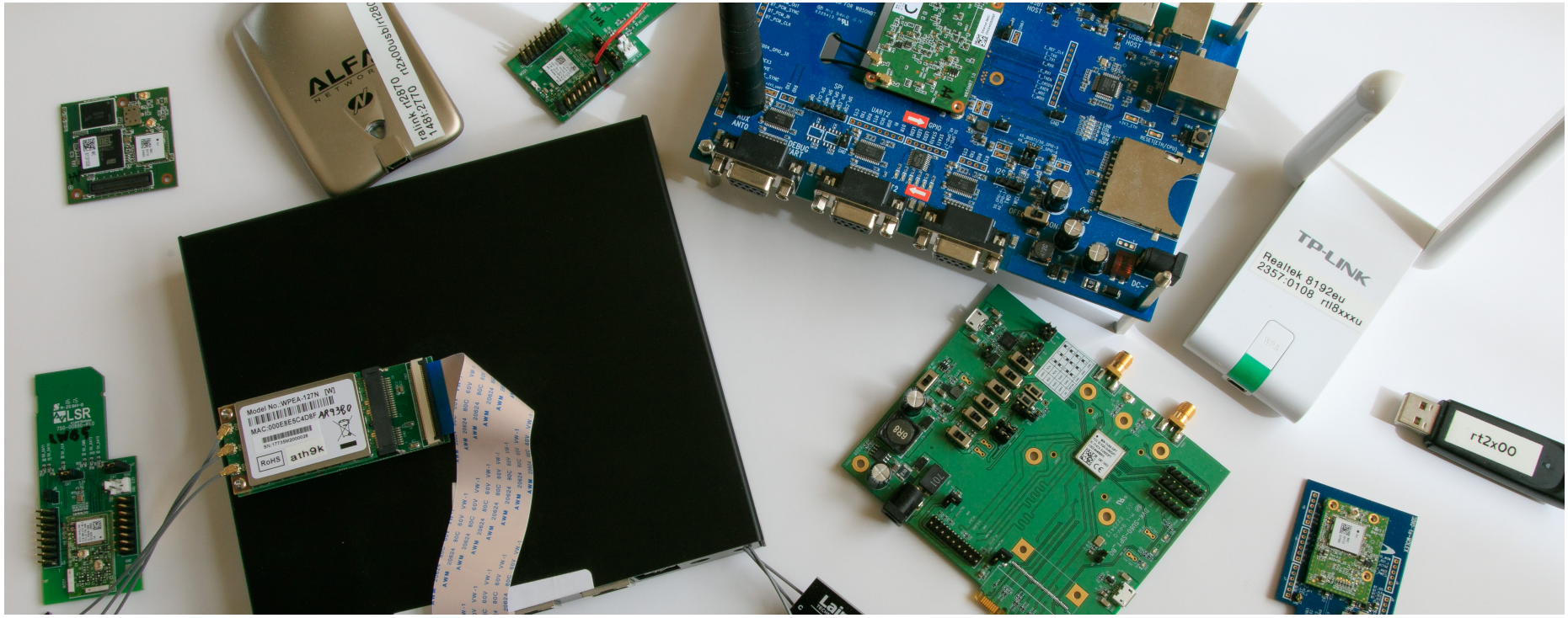
Linux wireless stack (from bottom up)

- Chip and bus
- Bus driver
- WiFi device driver
- mac80211 (Soft-MAC)
- cfg80211
- nl80211
- User-space management
- Data

- From the data-application side, no different than Ethernet or other systems
- Open UDP or TCP sockets

CSC
CAL-SIERRA
CONSULTING

Interfacing



CAL-SIERRA CONSULTING

Interfacing

- So you've got a new device, what do you do?
  - Plug it in, get the device IDs from the relevant bus tools
  - Determine what driver it matches
    - hotplug loads it or modprobe if necessary
  - If none, maybe it's similar to another chip?
    - Add the bus device IDs to the existing driver
    - Write from scratch?!? (call me!)
  - Get the basics working first, then move on to the other features

CSC
CAL-SIERRA
CONSULTING

Device IDs

- `lspci -nn` : "01:00.0 Network controller [0280]: Qualcomm Atheros AR93xx Wireless Network Adapter [168c:0030] (rev 01)"
- `lsusb -v` : "Bus 002 Device 002: ID 148f:2770 Ralink Technology, Corp. RT2770 Wireless Adapter"
- sdio, use sysfs:

```
# cd /sys/bus/sdio/devices/mmc0:0001:1
# cat vendor
0x0271
# cat device
0x0301
```

CAL-SIERRA CONSULTING

Device IDs in drivers

- Find and add to code as necessary
  Example: ath6kl, AR6003 0x0271:0x301

```
38  #define MANUFACTURER_ID_AR6003_BASE        0x300
39  #define MANUFACTURER_ID_AR6004_BASE        0x400
40      /* SDIO manufacturer ID and Codes */
41  #define MANUFACTURER_ID_ATH6KL_BASE_MASK     0xFF00
42  #define MANUFACTURER_CODE                  0x271        /* Atheros */
```

drivers/net/wireless/ath/ath6kl/hif.h

```
static const struct sdio_device_id ath6kl_sdio_devices[] = {
        {SDIO_DEVICE(MANUFACTURER_CODE, (MANUFACTURER_ID_AR6003_BASE | 0x0))},
        {SDIO_DEVICE(MANUFACTURER_CODE, (MANUFACTURER_ID_AR6003_BASE | 0x1))},
        {SDIO_DEVICE(MANUFACTURER_CODE, (MANUFACTURER_ID_AR6004_BASE | 0x0))},
        {SDIO_DEVICE(MANUFACTURER_CODE, (MANUFACTURER_ID_AR6004_BASE | 0x1))},
        {SDIO_DEVICE(MANUFACTURER_CODE, (MANUFACTURER_ID_AR6004_BASE | 0x2))},
        {},
};
```

drivers/net/wireless/ath/ath6kl/sdio.c

CAL-SIERRA CONSULTING

Good result

```
[   15.240000] mmc0: queuing unknown CIS tuple 0x01 (3 bytes)
[   15.250000] mmc0: queuing unknown CIS tuple 0x1a (5 bytes)
[   15.250000] mmc0: queuing unknown CIS tuple 0x1b (8 bytes)
[   15.260000] mmc0: queuing unknown CIS tuple 0x14 (0 bytes)
[   15.260000] mmc0: queuing unknown CIS tuple 0x80 (1 bytes)
[   15.260000] mmc0: queuing unknown CIS tuple 0x81 (1 bytes)
[   15.260000] mmc0: queuing unknown CIS tuple 0x82 (1 bytes)
[   15.260000] mmc0: new high speed SDIO card at address 0001
[   15.320000] ath6kl_sdio mmc0:0001:1: Direct firmware load for ath6k/AR6003/hw2.1.1/fw-5.bin failed with error -2
[   15.360000] ath6kl: initializing atheros generic netlink family
[   15.580000] random: nonblocking pool is initialized
[   16.130000] ath6kl: ar6003 hw 2.1.1 sdio fw 3.4.0.0094 api 4
[   16.130000] ath6kl: firmware supports: sched-scan,sta-p2pdev-duplex,rsn-cap-override,sscan-match-list,rssi-scan-thold,
```

CSC
CAL-SIERRA
CONSULTING

Bad result

- Is it plugged in correctly?
- Is it powered (sometimes a power enable pin)?
- Does it announce on bus?
- Does it fail to load firmware?
- Does it come up and look OK, but fails on `ifconfig wlan0 up`?
- Does it work OK, but won't connect to an open AP?
- Does it connect to open AP but other features won't work?
- Kill NetworkManager (or equivalent)

CSC
CAL-SIERRA
CONSULTING

A hunch is creativity trying to tell you something

CAL-SIERRA CONSULTING

Debugging tools

- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark

CAL-SIERRA CONSULTING

Debugging tools

- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark

- By default, debug config options are disabled
- brcmfmac: CONFIG_BRCMDBG
- ath6kl: ATH6KL_DEBUG, ATH6KL_TRACING

CSC
CAL-SIERRA
CONSULTING

Debugging tools

- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark

- Lots of debug printing to kernel log.
- Need to ++ debug print level and enable specific messages.
- ath6kl:

```
`echo 0x00140000 > /sys/module/
ath6kl_core/parameters/debug_mask`
```

CAL-SIERRA CONSULTING

Debugging tools

- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark

- Different drivers have special module parameters. brcmfmac:
  - debug - Sets debug level. The levels are defined in debug.h. Maintainer asks for kernel logs with debug level set to 0x1416, which shows driver-firmware interactions.
  - feature_disable - Override feature detection to avoid its use in driver.
  - ignore_probe_fail - Allow post-mortem debugging if firmware crash happens during probe. Check the forensics file in debugfs.

CSC
CAL-SIERRA
CONSULTING

Debugging tools

- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark

- Best place to check for existence, can walk the bus tree
- Shows you bound driver, etc

CSC
CAL-SIERRA
CONSULTING

Debugging tools

- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark

```
/sys/kernel/debug/ieee80211/phy0/ath6kl
# ls
bgscan_interval       fwlog                 reg_addr
create_qos            fwlog_block           reg_dump
credit_dist_stats     fwlog_mask            reg_write
delete_qos            keepalive             roam_mode
disconnect_timeout    listen_interval       roam_table
endpoint_stats        lrssi_roam_threshold  tgt_stats
force_roam            power_params          war_stats
```

- brcmfmac:
  - revinfo - revision of h/w and firmware
  - features - firmware features
  - msgbuf_stats (pcie only) - stats counters of msgbuf layer
  - counters (sdio) - stats counter of sdio bus layer
  - console_interval (sdio) - period to obtain fw console content
  - forensics (sdio) - dump fw console and trap info

CAL-SIERRA CONSULTING

Debugging tools

- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark

- Many drivers will dump firmware crashes to sysfs “file”
- brcmfmac: /sys/class/devcoredump

CAL-SIERRA CONSULTING

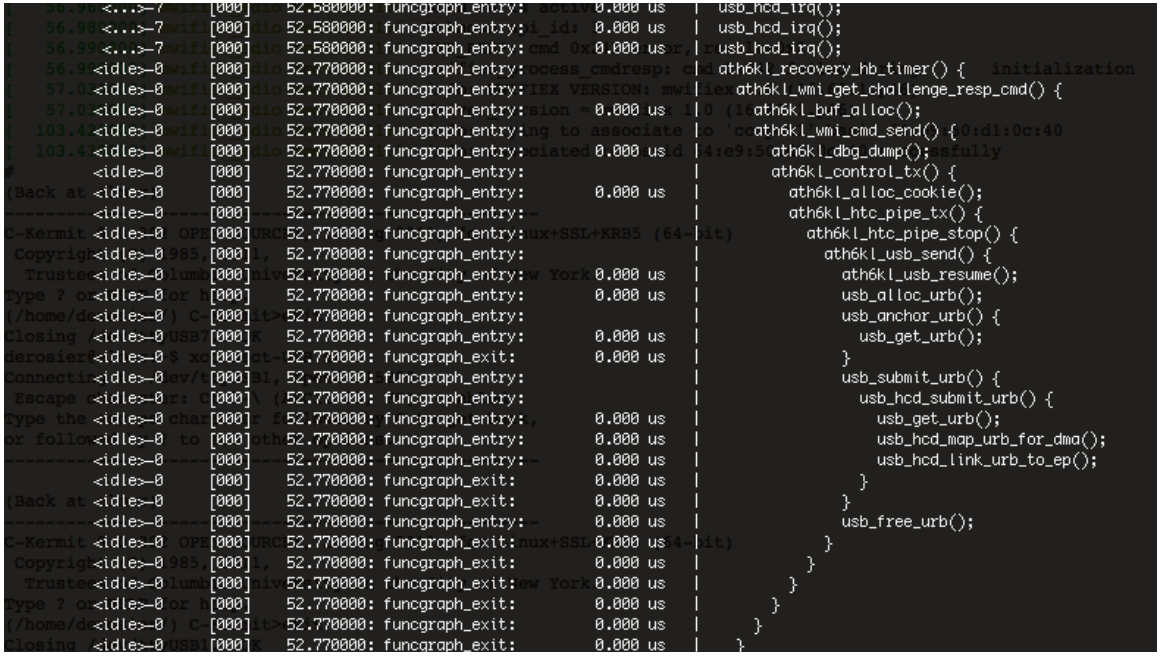
Debugging tools

- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark

- Kernel's function tracer
- Must enable
- Useful to dig through call-stacks and figure out what's going on live
- Also can profile the stack

CSC
CAL-SIERRA
CONSULTING

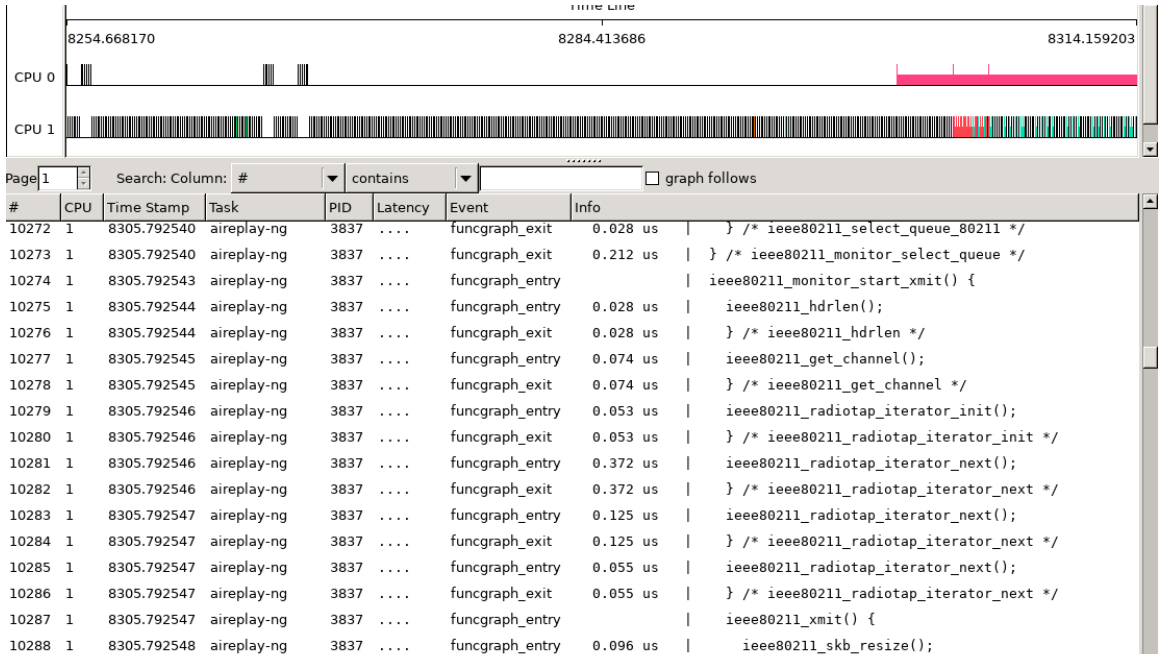
ftrace



trace-cmd

CAL-SIERRA CONSULTING

ftrace



#	CPU	Time Stamp	Task	PID	Latency	Event		Info
10272	1	8305.792540	aireplay-ng	3837	....	funcgraph_exit	0.028 us	} /* ieee80211_select_queue_80211 */
10273	1	8305.792540	aireplay-ng	3837	....	funcgraph_exit	0.212 us	} /* ieee80211_monitor_select_queue */
10274	1	8305.792543	aireplay-ng	3837	....	funcgraph_entry		ieee80211_monitor_start_xmit() {
10275	1	8305.792544	aireplay-ng	3837	....	funcgraph_entry	0.028 us	ieee80211_hdrlen();
10276	1	8305.792544	aireplay-ng	3837	....	funcgraph_exit	0.028 us	} /* ieee80211_hdrlen */
10277	1	8305.792545	aireplay-ng	3837	....	funcgraph_entry	0.074 us	ieee80211_get_channel();
10278	1	8305.792545	aireplay-ng	3837	....	funcgraph_exit	0.074 us	} /* ieee80211_get_channel */
10279	1	8305.792546	aireplay-ng	3837	....	funcgraph_entry	0.053 us	ieee80211_radiotap_iterator_init();
10280	1	8305.792546	aireplay-ng	3837	....	funcgraph_exit	0.053 us	} /* ieee80211_radiotap_iterator_init */
10281	1	8305.792546	aireplay-ng	3837	....	funcgraph_entry	0.372 us	ieee80211_radiotap_iterator_next();
10282	1	8305.792546	aireplay-ng	3837	....	funcgraph_exit	0.372 us	} /* ieee80211_radiotap_iterator_next */
10283	1	8305.792547	aireplay-ng	3837	....	funcgraph_entry	0.125 us	ieee80211_radiotap_iterator_next();
10284	1	8305.792547	aireplay-ng	3837	....	funcgraph_exit	0.125 us	} /* ieee80211_radiotap_iterator_next */
10285	1	8305.792547	aireplay-ng	3837	....	funcgraph_entry	0.055 us	ieee80211_radiotap_iterator_next();
10286	1	8305.792547	aireplay-ng	3837	....	funcgraph_exit	0.055 us	} /* ieee80211_radiotap_iterator_next */
10287	1	8305.792547	aireplay-ng	3837	....	funcgraph_entry		ieee80211_xmit() {
10288	1	8305.792548	aireplay-ng	3837	....	funcgraph_entry	0.096 us	ieee80211_skb_resize();

kernelshark

CAL-SIERRA CONSULTING

Debugging tools

- kconfig debug options
- dmesg
- module parameters
- sysfs
- debugfs
- coredump
- ftrace
- Wireshark

- Device is working, but actual communication fails
- Do the packets go on the air?
- Is there something wrong with what's going on?

CAL-SIERRA CONSULTING

Wireshark

```
fail_carl9170_02.
.i02:/home/derosier/projects/aircrack-ng-tr

(OCSIWMODE) failed: Device or resource busy
   Trying broadcast probe requests...
   Try 0, from MAC: 0:9d:85:7a:b2:b
   Try 0, timeout
   Try 1, from MAC: 0:1a:b0:66:4d:e5
   Try 0, timeout
   Try 2, from MAC: 0:37:20:9a:e2:c2
   Try 0, timeout
   No Answer...
   Found 6 APs

   Trying directed probe requests...
   84:00:2D:E1:C8:88 - channel: 1 - ''
    0/30:    0%

   84:00:2D:E1:C8:8B - channel: 1 - ''
    0/30:    0%

   84:00:2D:E1:C8:8D - channel: 1 - ''
    0/30:    0%

   C0:7C:D1:27:71:18 - channel: 1 - 'HOME-8
    0/30:    0%

   84:00:2D:E1:C8:8A - channel: 1 - 'xfinit
    0/30:    0%

   60:38:E0:0F:F3:00 - channel: 11 - 'cshom
```

fail_carl9170_02.pcapng

wlan.fc.type_subtype == 0x0004 || wlan.fc.type_subtype == 0x0005

No.	Time	Source	Destination	Protocol	Length	Info
149	4.559862932	00:9d:85:7a:b2:0b	Broadcast	802.11	60	Probe Request, SN=0, FN=0, Flags=........, SSI
173	5.160094019	SignalNe_66:4d:e5	Broadcast	802.11	60	Probe Request, SN=1, FN=0, Flags=........, SSI
174	5.162406280	Pegatron_e1:c8:8a	SignalNe_66:4d:e5	802.11	196	Probe Response, SN=2472, FN=0, Flags=........,
175	5.164131668	Pegatron_e1:c8:8a	SignalNe_66:4d:e5	802.11	196	Probe Response, SN=2472, FN=0, Flags=....R...,
199	5.815184129	00:37:20:9a:e2:c2	Broadcast	802.11	60	Probe Request, SN=2, FN=0, Flags=........, SSI
200	5.817498107	Pegatron_e1:c8:8a	00:37:20:9a:e2:c2	802.11	196	Probe Response, SN=2475, FN=0, Flags=........,
201	5.819294214	Pegatron_e1:c8:8a	00:37:20:9a:e2:c2	802.11	196	Probe Response, SN=2475, FN=0, Flags=....R...,
202	5.821043580	Pegatron_e1:c8:8a	00:37:20:9a:e2:c2	802.11	196	Probe Response, SN=2475, FN=0, Flags=....R...,
203	5.822794892	Pegatron_e1:c8:8a	00:37:20:9a:e2:c2	802.11	196	Probe Response, SN=2475, FN=0, Flags=....R...,
221	6.360395632	00:8d:6c:2c:33:d7	Broadcast	802.11	60	Probe Request, SN=3, FN=0, Flags=........, SSI
227	6.367257749	Pegatron_e1:c8:8a	00:8d:6c:2c:33:d7	802.11	196	Probe Response, SN=2478, FN=0, Flags=....R...,
237	6.561287405	00:65:94:20:2f:03	Broadcast	802.11	60	Probe Request, SN=7, FN=0, Flags=........, SSI
248	6.764723646	MS-NLB-VirtServer_…	Broadcast	802.11	60	Probe Request, SN=11, FN=0, Flags=........, SS

```
▶ Frame 199: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
▶ Radiotap Header v0, Length 18
▼ 802.11 radio information
    PHY type: 802.11b (4)
    Short preamble: False
    Data rate: 1.0 Mb/s
    Channel: 1
    Frequency: 2412MHz
    Signal strength (dBm): -32dBm
  ▶ [Duration: 528µs]
▼ IEEE 802.11 Probe Request, Flags: ........
    Type/Subtype: Probe Request (0x0004)
  ▶ Frame Control Field: 0x4000
    .000 0000 0000 0000 = Duration: 0 microseconds
    Receiver address: Broadcast (ff:ff:ff:ff:ff:ff)
    Destination address: Broadcast (ff:ff:ff:ff:ff:ff)
    Transmitter address: 00:37:20:9a:e2:c2 (00:37:20:9a:e2:c2)
    Source address: 00:37:20:9a:e2:c2 (00:37:20:9a:e2:c2)
0000  00 00 12 00 2e 48 00 00  00 02 6c 09 a0 00 e0 01   .....H.. ..l.....
0010  00 00 40 00 00 00 ff ff  ff ff ff ff 00 37 20 9a   ..@..... .....7 .
0020  e2 c2 ff ff ff ff ff ff  20 00 00 00 01 04 02 04   ........ ........
```

CAL-SIERRA CONSULTING

Getting help

- linux-wireless list - need more than “it doesn’t work”:
  - enable debugging features
  - send kernel logs with those debugging on
  - be specific about hardware, device, firmware version, Linux version etc.
  - be specific about what doesn’t work and what you’ve tried
  - expect to include wireless captures

CSC
CAL-SIERRA
CONSULTING

Links

- Linux-wireless wiki: https://wireless.wiki.kernel.org/
- Linux-firmware: https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmware.git/
- Linux wireless mailing list: https://wireless.wiki.kernel.org/en/developers/mailinglists
- Help with iwlwifi: https://wireless.wiki.kernel.org/en/users/drivers/iwlwifi/debugging

CSC
CAL-SIERRA
CONSULTING

CAL-SIERRA
CONSULTING

STEVE DEROSIER

steve@cal-sierra.com
+1-916-203-5695